

Laboratorio II

Trabajo Práctico 3

Henestroza, Alejandro Iván

Índice

1. [Introducción](#)
2. [Interfaz Gráfica](#)
 1. [Generar Orden](#)
 2. [Listar Órdenes](#)
 3. [Borrar Órdenes](#)
 4. [Modificar Stock](#)
 5. [Importar Listado de Órdenes](#)
 6. [Exportar Listado de Órdenes](#)
 7. [Guardar Listado a Archivos](#)
3. [Entidades](#)
4. [Tests Unitarios](#)
5. [Test Aplicación de Consola](#)
6. [Dónde encontrar cada tema visto en la cursada](#)

Introducción

Se ha desarrollado una aplicación que permite generar órdenes de producción de bicicletas y accesorios.

Hay dos tipos de bicicletas, Mountain Bike y Playera, que pueden ordenarse, así como también dos tipos de accesorios, Casco y Luz.

Las bicicletas están compuestas de dos ruedas y un cuadro, el cual está fabricado de un material. La aplicación sigue el stock de material y en base al mismo, permite la generación de nuevas órdenes. También permite modificar el stock, agregando o quitando material.

Finalmente, la aplicación permite exportar cada orden a un archivo de texto separado, con los datos necesarios para ser fabricada. Los archivos de texto se guardan en la carpeta Ordenes, ubicada en la misma carpeta que contiene a la solución. También permite exportar el listado de órdenes en formato binario e importar un listado. Se puede exportar con el nombre que se desee, en el directorio deseado.

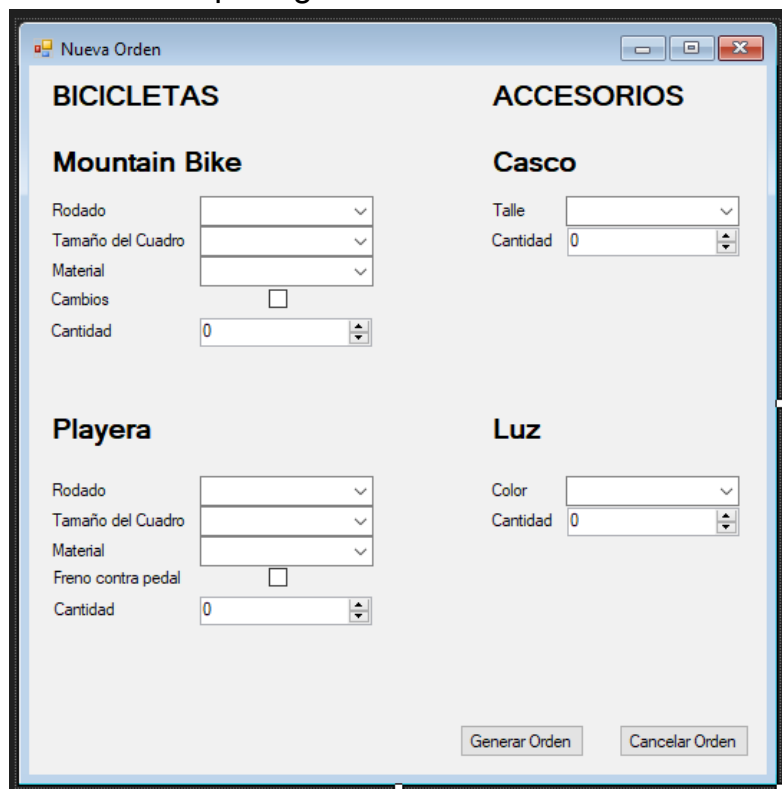
Interfaz Gráfica

La interfaz gráfica consiste en cuatro formularios. El formulario principal contiene los botones para cada funcionalidad de la aplicación.



1: Generar Orden

Este botón abre el formulario para generar una orden.



Se pueden agregar en conjunto o de forma individual. El formulario solo añadirá a la orden los elementos para los cuales se completen todos sus campos. Es decir, se puede ordenar únicamente bicicletas playeras, o bicicletas playeras y cascos.

2: Listar Órdenes

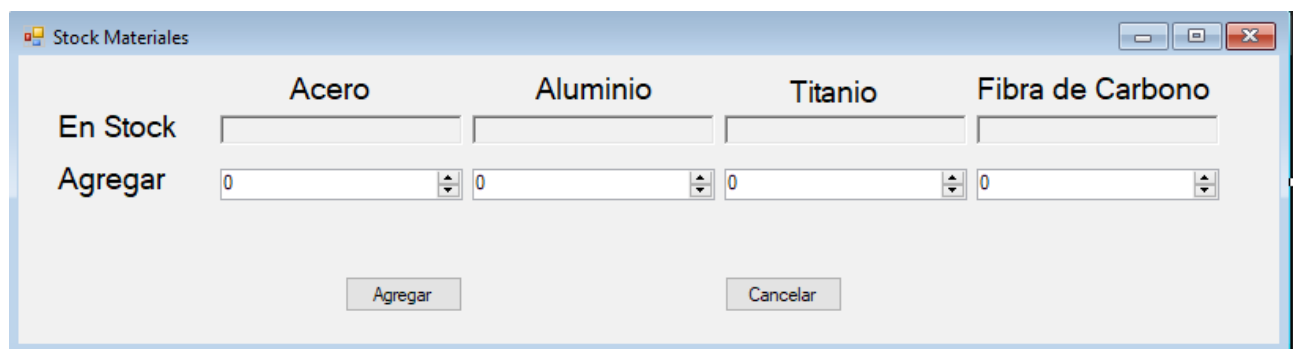
Muestra el listado de órdenes, si es que hay uno. Si no se ha creado ninguna orden o no se ha importado ningún listado, no se podrá abrir este formulario.

3: Borrar Órdenes

Este botón eliminará el listado de órdenes (en caso de que haya uno cargado).

4: Modificar Stock

Este formulario mostrará el stock disponible de cada material, permitiendo agregar stock. Antes de generar una orden de bicicletas, se debe añadir stock (inicialmente es 0 para cada material).



	Acero	Aluminio	Titanio	Fibra de Carbono
En Stock	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Agregar	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>

Los valores máximos son 100.000. También se pueden pasar valores negativos (hasta -100.000), para remover stock.

Los rich text box se cargan con el stock disponible al momento de abrir el formulario.

5: Importar Listado de Órdenes

Este botón abrirá un cuadro de diálogo para seleccionar el archivo a importar. Solo permite seleccionar archivos con la extensión .bin.

6: Exportar Listado de Órdenes

Este botón abrirá un cuadro de diálogo para seleccionar la ubicación del archivo a exportar, así como también permitirá definir su nombre. Sólo extensión .bin.

7: Guardar Listado a Archivos

Este botón tomará el listado cargado (en caso de que exista) y creará un archivo .txt con el nombre "Orden [número de orden]". El número de orden corresponde a su índice en el listado. Los archivos se guardan en la carpeta Órdenes, ubicada en el directorio que contenga el archivo .sln.

Entidades

Bicicleta

Esta es la clase abstracta y genérica base de la cual heredan Mountain Bike y Playera. Recibe un parámetro de clase de tipo Material, el cual será utilizado para inicializar un objeto de tipo Cuadro correspondiente a la bicicleta que se esté instanciando.

Implementa la interfaz *IFabricable*, que define dos métodos: *EsFabricable* y *CalcularCosto*. La clase Bicicleta implementa ambos métodos. Para el primero, valida que los parámetros recibidos por el constructor de Bicicleta sean válidos. El segundo retorna el costo de la bicicleta (que incluye el costo de sus componentes).

MountainBike

Hereda de Bicicleta, define un campo adicional *TieneCambios*. En base a este campo, su costo puede ser mayor o menor.

Playera

Hereda de Bicicleta, define un campo adicional *FrenoContraPedal*. En base a este campo, su costo puede ser mayor o menor.

Material

Esta es la clase abstracta que define los campos de cada material (nombre y costo). Los materiales que hereden de esta clase deben implementar un constructor vacío para ser utilizados en la fabricación de una Bicicleta, debido a las restricciones que le impone a su tipo genérico.

Las clases derivadas de *Material* deben llamar al constructor base, pasando por parámetros el nombre del material y el costo.

Los costos de los materiales ya existentes se encuentran en el enumerado *EnumCostoMaterial*.

Accesorio

Esta es la clase abstracta de la cual heredan Casco y Luz. Define los campos esenciales (nombre y costo).

Las clases derivadas incluyen nuevos campos, en base a los cuales podrán modificar el costo del accesorio.

Cuadro

Esta clase define un componente de Bicicleta. El cuadro debe recibir una instancia de Material en su constructor, la cual se obtiene a partir del parámetro genérico de la clase Bicicleta (y sus derivadas).

Rueda

Esta clase define un componente de Bicicleta. Recibe el tamaño del rodado en su constructor y en base al mismo, establece el costo de la rueda.

Orden

Esta clase define una orden que se enviará a producción. Contiene un listado de objetos que implementen la interfaz *Ifabricable* y un listado de *Accesorio*.

Debido a la estructura de la aplicación, esta clase se encargará de instanciar las Bicicletas y Accesorios a producir (el proyecto de formulario accede a esta clase). Define los métodos para agregar múltiples bicicletas y accesorios de cada tipo.

ListOrdenSerializable

Esta clase define un listado de órdenes que puede ser serializable a formato binario. Define también el método para imprimir cada orden a un archivo de texto.

Fabrica

Esta clase estática será accesible por el proyecto de formulario. La misma tendrá un *ListOrdenSerializable*, así como también controlará el stock de materiales disponible.

Esta clase envuelve los métodos de AgregarOrden, ImprimirArchivos, correspondientes a Orden y *ListOrdenSerializable*, para que los mismos sean accedidos a través de Fábrica y no directamente de sus clases.

Define también los métodos para importar y exportar el listado de órdenes.

Tests Unitarios

Se han definido dos tests unitarios. Uno para comprobar el funcionamiento de la clase genérica *Bicicleta* (testeando una de sus clases derivadas) y otro para comprobar el funcionamiento de la clase *Orden*.

Se ha comprobado que no se permita instanciar *Bicicletas* con parámetros inválidos (ruedas de distinto tamaño) o que se instancien con rodados no admitidos por el tipo de *Bicicleta*.

En cuanto a *Orden*, se ha testeado que pueda agregar múltiples *Bicicletas* y *Accesorios*.

Test Aplicación de Consola

Se ha generado un proyecto de consola en el cual se ha comprobado el funcionamiento de la serialización. Se generan dos órdenes, se agregan al *ListOrdenSerializable* que contiene la clase estática *Fabrica* y se ha exportado el listado a binario. Luego se limpia el listado, comprobando que efectivamente se haya eliminado la referencia al anterior. Finalmente, se vuelve a importar el listado original y se comprueba que las órdenes se hayan cargado correctamente, imprimiendo las mismas en archivos de texto.

Dónde encontrar cada tema visto en la cursada

Excepciones

Se han implementado dos excepciones propias:

- `FechaInvalidaException`
- `RuedaIncompatibleException`

`FechaInvalidaException` controla que la fecha estimada para la producción de una orden sea posterior a su fecha de creación. Esta excepción, si bien está controlada en el constructor de la clase `Orden`, no es utilizada en el proyecto final, dado que el formulario de creación de `Orden` aún no permite establecer una fecha de producción.

No obstante, se ha testado la misma en el proyecto de Test Unitario `TestEntidades`, en la clase `TestOrden`, para comprobar que la excepción se lance en el momento apropiado.

`RuedaIncompatibleException` controla que no se instancie una `Bicicleta` con ruedas inválidas. Hay dos situaciones que califican para esta excepción:

1. Que las ruedas sean de tamaño distinto
2. Que la bicicleta no admita el rodado (por ejemplo, `MountainBike` solo puede recibir ruedas de rodado 26, 27 o 28)

Tipos Genéricos

La clase `Bicicleta` (y sus derivadas) utiliza tipos genéricos para definir el material del cuadro de la `Bicicleta`. Se han aplicado dos restricciones: el tipo debe ser `Material` (o, al ser una clase abstracta, una de sus derivadas) y debe implementar un constructor sin parámetros. Por esta razón, todos los materiales deben implementar un constructor vacío, que llame al constructor de la clase base `Material`.

En la clase `Orden` se han creado métodos que reciben un tipo genérico para agregar `Bicicletas` a la `Orden`. Esto es porque la `Orden` llama al constructor de `Bicicleta`, debiendo pasar el tipo genérico al hacerlo.

Interfaces

Se ha creado una interfaz `IFabricable`, la cual define dos métodos: *`EsFabricable`* y *`CalcularCosto`*.

Esta interfaz no solo define estos métodos, sino que también es utilizada (no implementada) en la clase `Orden`. `Orden` define un listado de productos que implementen la interfaz *`IFabricable`*, para así poder utilizar el método *`CalcularCosto`* implementado por `Bicicleta`.

Archivos

La clase *ListOrdenSerializable* y la clase estática *Fabrica* implementan los dos temas vistos: Streams y Serialización.

Fabrica es la clase que se encarga de serializar a *ListOrdenSerializable* (que es la clase serializable). Por su parte, *ListOrdenSerializable*, al contener un listado de Órden, utiliza *StreamWriter* para crear archivos de texto por cada Orden que contenga.